Administration and audit manual

Guide to auditing the console: authentication, session, authorisation, log integrity, header, infrastructure and code review controls, with their severity criterion.

Audit manual — Abzu Console (demo V-2)

Document version: 2026-10-16 · Console V-2 (post-remediation) · Language: English

Document for whoever must **audit** the Abzu console: what to verify, where each piece of evidence is, and how to reproduce each control. It reflects the corrected state after the red-team audit of 2026-10-16 (4 tracks, OSCP methodology) and its remediation.

1. System map

Component	Description	Notes
Source code	Product repository	No base-technology marks on the public surface
Console	Installation web service	Access over HTTPS
Agent	Agent binary on each managed server	Outbound channel with mutual authentication
Secrets	Installation secret store, with restrictive permissions	Never print values
Data	Audit log and installation state	Restricted access
Services	Operating system console and agent services	Dedicated service accounts

Access for auditing is performed with the authorised credentials and procedures of the installation itself.

2. Controls checklist

2.1 Authentication

- [ ] **Generic error**: login with a bad password (SEGRD, government or invalid CSRF) always returns `/?error=credenciales`, without distinguishing the cause.
  - `curl -s -o /dev/null -w "%{http_code} %{redirect_url}\n" --data-urlencode "username=Ogtic" --data-urlencode "password=MALA" the console sign-in path` (with fresh CSRF from `GET /?via=gobierno`)
- [ ] **Rate limiting**: 10 attempts / 5 min per IP → `429` + `login_rate_limited` event in audit.

- [ ] **Identity provider fail-closed**: if the SEGRD provider denies or fails technically, there is NO automatic fallback to local access; the technical error shows a 503 page with an explicit link to `?via=gobierno` and records `login_proveedor_degradado`.
- [ ] **The password is never persisted**: search for the word "password" in the log → 0 matches.
- [ ] **WebAuthn**: `user_verification=REQUIRED` at login (authentication options and response verification); clone detection by `sign_count`; RP ID/origin from the installation configuration at publication.

2.2 Sessions and CSRF

- [ ] Cookie `abzu_sess`: `HttpOnly`, `SameSite=Lax`, `Secure` when `COOKIE_SECURE=1`; single helper `_make_cookie()` (also in the passkey flow).
- [ ] TTL 4 h + idle 60 min (`last_seen` sliding); `tenant` bound to the session and validated against `?tenant=`.
- [ ] Single-use CSRF at login, logout (POST), `credentials/delete` and `register/*`; purge of expired tokens.
- [ ] Logout invalidates on the server: after `POST /auth/logout`, `/api/me` → 401 with the old cookie.
- [ ] No session fixation: the sid is generated by the server (`secrets.token_urlsafe(32)`), HMAC-SHA256 signed with `compare_digest`.

2.3 Authorisation and tenants

- [ ] No session → 401 on all `/api/*` (test the 7 endpoints).
- [ ] `?tenant=<nonexistent-GUID>` → 404 on `/` and `/consola`; session tenant vs URL mismatch → 404.
- [ ] Sensitive reads (`/api/status`, `/api/backups`, `/api/audit`) leave an `api_read` event in the log.

2.4 Audit log integrity

- [ ] Events recorded: `login_*_ok/fail`, `login_proveedor_degradado`, `login_rate_limited`, `login_webauthn_*`, `webauthn_register`, `logout`, `api_read`.
- [ ] The `username` is sanitised (max. 128 chars) before recording; the IP is taken from the connection except for explicit `TRUSTED_PROXIES` (no blind trust in `X-Forwarded-For`).
- [ ] The log is append-only (`api/audit.jsonl`, 0640); verify that no endpoint allows modifying it.

2.5 HTTP headers and surface

- [ ] `X-Frame-Options: DENY`, `Content-Security-Policy` (with `frame-ancestors 'none'`), `X-Content-Type-Options: nosniff`, `Referrer-Policy: no-referrer`; HSTS only under HTTPS.
- [ ] Generic `Server: Abzu` (without environment version).

- [ ] `PUT/DELETE/PATCH/OPTIONS` → 501; `/static/../` → 404 (anti-traversal); `_read_form` capped at 256 KB → 413 if exceeded.

2.6 Infrastructure

- [ ] Services run as a dedicated service user (not as system administrator).
- [ ] Service units with hardening: dedicated user, no new privileges, protected file system and private temporary directories.
- [ ] No secrets in the process environment: secrets are read from their store with restrictive permissions and the process environment does not expose secret names.
- [ ] Code and data with service ownership and restrictive permissions; the access credential store with permission only for its owner.
- [ ] No deviation: the fingerprint of the deployed code matches that of the repository and the working tree is clean.
- [ ] Privilege configuration without extraneous exceptions; logs with restrictive permissions; the system journal does not leak container inventory.

2.7 Code (static review)

- [ ] No `random` (only `secrets`), no command interpreter invocation or direct queries; `compare_digest` in sensitive comparisons.
- [ ] `esc()` / `html.escape` on all dynamic data; no `eval` / `document.write`; no tokens in `localStorage`.
- [ ] `git log -p --all` without secrets (only `<redacted>` placeholders in docs).
- [ ] Agent: TLS with active verification; unverified mode is documented as prohibited in production.

3. How to reproduce the 2026-10-16 audit

It was executed in 4 parallel tracks (OSCP methodology, grey box, read-only / non-destructive exploitation):

Track	Focus	Result
1. Web pentest	Logins, sessions, CSRF, tenant GUID, injection, errors	0 critical/high; 3 medium (Secure cookie, RP ID by Host, CSP)
2. API	`/api/*` inventory, authorisation, audit log integrity	2 high (no rate limit; identity provider fail-open), 4 medium, 5 low
3. Infra/ config	systemd, permissions, secrets, network, agent	2 high (root without hardening; secrets in environ), 5 medium, 4 low
4. Code review	console code, interface and agent	2 high (fail-open; no user verification), 6 medium, 8 low

Remediation applied (console V-2, deployed and verified): identity provider fail-closed, rate limiting 10/5min, dedicated service user + unit hardening, secrets out of environ,

`user_verification=REQUIRED`, relying party identifier and origin from configuration, centralised cookie with `Secure`, CSRF on WebAuthn endpoints, provider token in HttpOnly cookie (out of the URL), 256 KB cap on forms, XFF only from trusted proxies, CSP + `X-Frame-Options: DENY` + HSTS, sessions 4 h + idle 60 min + tenant bound, CSRF purge, audit of API reads, logout by POST, honest mTLS text in UI.

4. Accepted / deferred risks (not open findings)

- **Rebuild of the agent binary** with the current toolchain: separate task on the build server.
- **Agent mTLS enrolment**: pending for v1 (today static token; rotation requires owner OK).
- **Rotation of object repository credentials / agent token**: recommended as hygiene, requires the responsible person's authorisation.
- **Update of the web authentication library**: monitor public security advisories.
- **Legacy OIDC flow**: dormant; assess its removal.
- **Complete WebAuthn ceremony**: pending HTTPS + physical authenticator.
- **In-memory rate limit**: lost on restart (acceptable for demo).

5. Severity criterion

- **Critical**: RCE, total authentication bypass, secret exfiltration.
- **High**: partial auth bypass, escalation, silent security degradation.
- **Medium**: hardening needed before exposing to clients (rate limit, CSRF, headers, scoping).
- **Low**: hygiene (versions, logs, texts, permissions not exploitable today).

6. Recommended frequency

Repeat this audit (the 4 tracks) **before each publication** (`abzu.segrd.com` or major update) and after any change in authentication, sessions or WebAuthn.